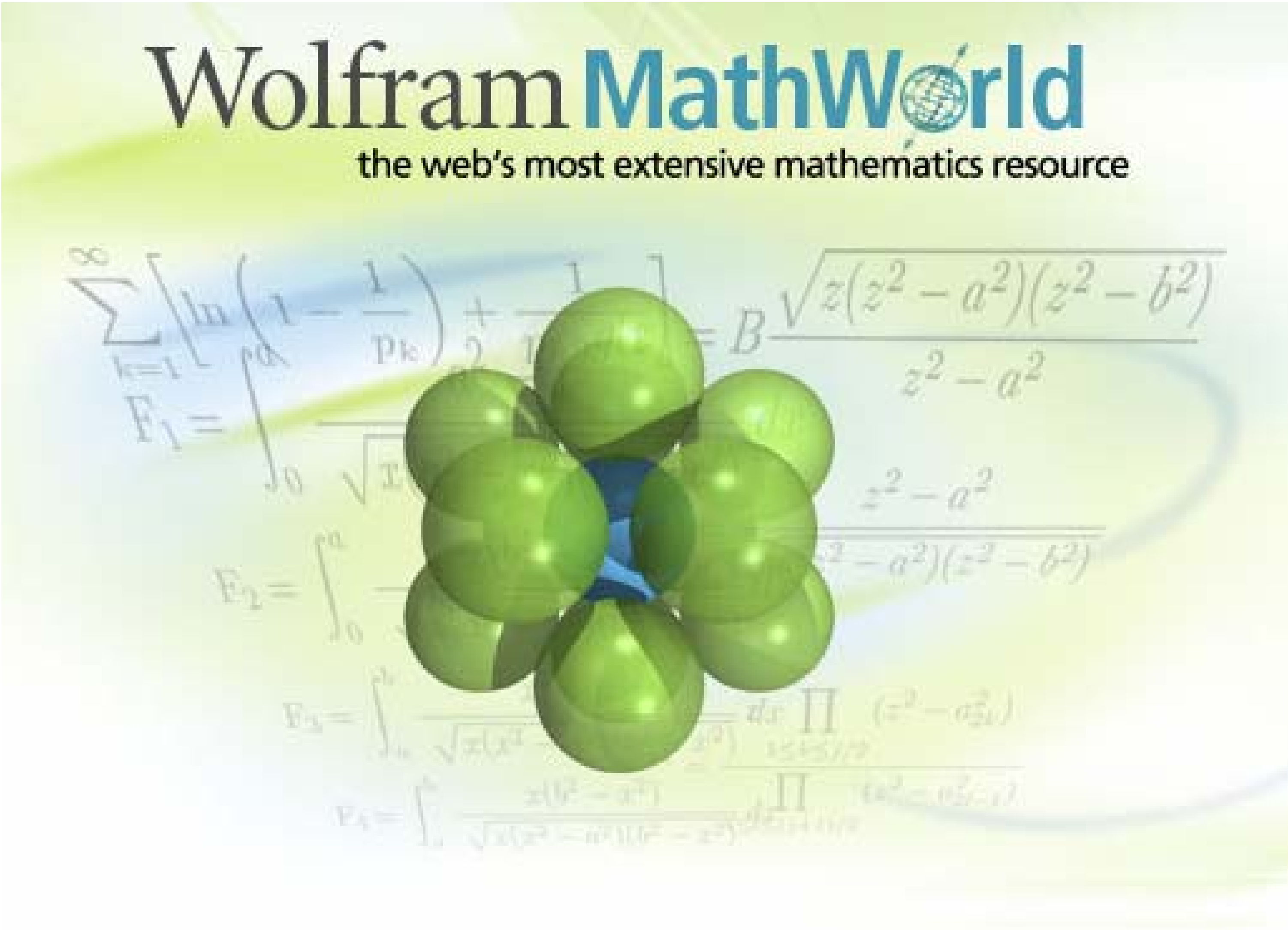
# 数学基础

(版本2009)

刘汝佳

# Mathworld.wolfram.com

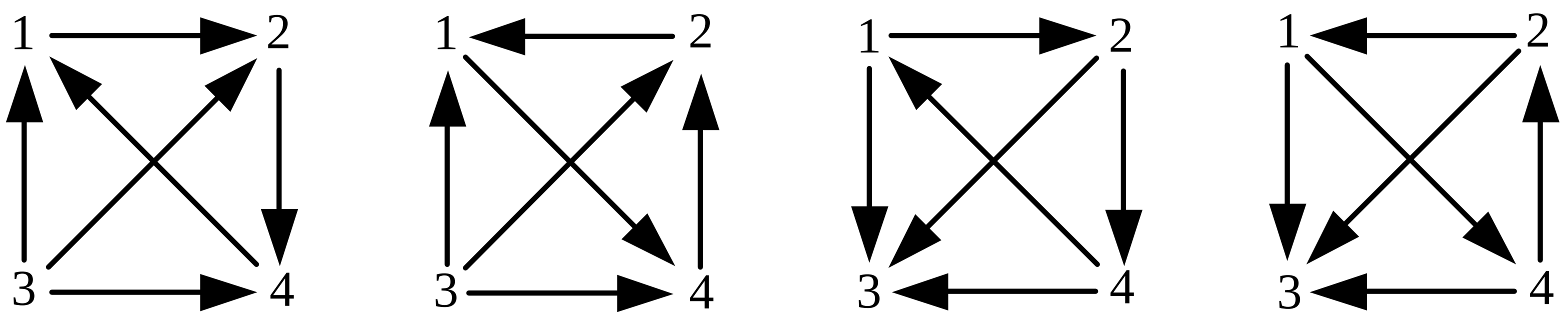


# 例1. 同构计数

- 一个竞赛图是这样的有向图
  - 任两个不同的点*u*、*v*之间有且只有一条边
  - 不存在自环
- 用*P*表示对竞赛图顶点的一个置换。当任两个不同顶点*u*、*v*间直接相连的边的方向与顶点*P*(*u*)、*P*(*v*)间的一样时，称该图在置换*P*下同构
- 对给定的置换*P*，我们想知道对多少种竞赛图在置换*P*下同构

# 例1. 同构计数

- 例：有顶点1、2、3、4和置换*P*：*P*(1)=2，*P*(2)=4，*P*(3)=3，*P*(4)=1
- 对于下图的四种竞赛图，在置换*P*下同构



# 分析

- 先把置换它分解成为循环, 首先证明长度为L的偶循环将导致无解
  - 对于点$i_1$, 记$P(i_k)=i_{k+1}$, 假设$i_1$和$i_{L/2+1}$的边方向为$i_1$->$i_{L/2+1}$, 那么有$i_2$->$i_{L/2+2}$, $i_3$->$i_{L/2+3}$, …, $i_{L/2+1}$->$i_1$, 和假设矛盾!
- 假设确定其中k条独立边后其他边也会确定, 则答案为$2^k$
- 考虑两类边: 循环内边和循环间边.

# 分析

- 循环内顶点的关系
  - 定了$i_1$和$i_j$之间的关系, $i_k$与$i_{(k+j-2) \bmod n+1}$之间的关系也被确定下来了, 因此只需要确定$i_1$和$i_2$, $i_3$, …, $i_{(L-1)/2+1}$这(L-1)/2对顶点的关系
- 不同循环间顶点的关系
  - 设循环为$(i_1,i_2,\ldots,i_{L1})$和$(j_1,j_2,\ldots,j_{L2})$, 通过类似分析得只需要确定gcd(L1, L2)对关系即可

# 分析

- 最后答案为$2^{k1+k2}$
- 其中k1=sum{(L-1)/2}, k2=sum{gcd(L1, L2)}
- 可以用二分法加速求幂

# 例2. 图形变换

- 平面上有n个点需要依次进行m个变换处理
- 规则有4种, 分别把$(x_0,y_0)$变为
  - 平移M(x, y): $(x_0+x, y_0+y)$
  - 缩放Z(L): $(Lx_0, Ly_0)$
  - 翻转F(0): $(x_0, -y_0)$; F(1): $(-x_0, y_0)$
  - 旋转R(a): a为正时逆时针, 离原点距离不变, 旋转a度
- 给n($\le 10^6$)个点和m($\le 10^6$)条指令
- 求所有指令完成后每个点的坐标

# 分析

- 如果直接模拟, 每次需要O(n)的时间, 一共O(nm), 无法承受
- 把点$(x_0, y_0)$写成列向量$[x_0, y_0]^T$, 则后3种变换可以都可以写成矩阵
  - 缩放P' = Z * P, Z = [L 0; 0 L]
  - 翻转P' = F * P, F = [1 0; 0 -1]或[-1 0; 0 1]
  - 旋转P' = R * P, R = [cosa –sina; sina cosa]
- 可是无法实现平移, 怎么办呢?

# 分析

- 修改表达方式，令P = $[x_0, y_0, 1]^T$, 则四种变换的矩阵M, Z, F, R分别为

$$M = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}, \quad Z = \begin{bmatrix} L & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad F = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{或} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$, R = \begin{bmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# 分析

- 只需要先计算所有变换矩阵的乘积A, 然后对于每个点, P’ = A * P
- 注意: 矩阵乘法不满足交换律, 因此需要按照输入顺序进行乘法
- 每次矩阵乘法需要$3^3$=27次乘法, 则计算A一共需要27m次乘法, 对所有点变换需要27n次乘法, 一共27(n+m)次

# 例3. 染色方案

- $N*M(N<=10^{100}, M<=5)$的格子纸，每个格子被填上黑色或者白色。求没有任何一个2*2的格子同色的染色方案总数 mod P。

# 分析

- 每行最多32(=$2^M$)种状态
- 任两种状态是否可组成相邻行，可以用一个32*32的矩阵S表示
- 下面证明$S^k$的元素(i,j)表示以i为第一层, j为最后一层, 共k+1层的方法数
- K=0时显然成立, 考虑$S^k=S^{k-1}*S$, 任何一个元素$S^K(i,j)=sum\{S^{K-1}(i,x)*S(x,j)\}$, 乘法原理
- 因此$S^n$的所有的元素之和就是答案

# 例4. 硬币

- 给定长度为K-1的二进制数组A，对K个排放好的硬币序列C，在其上定义X操作：
  - 将C向左循环移动一格
  - 如果第i（$1 \leqslant i < K$）个硬币背面朝上，并且Ai=1，那么将第K个硬币翻面。
- 给定L组，K个硬币的样板X操作情况（初始状态和X操作后的状态），保证它们可以唯一地得到A，并且一定能得到A。
- 另外有一个长度为N的硬币序列，在其上进行M组操作，每组操作对$S_i$~$S_i$+K的硬币进行$D_i$次X操作。给定这组硬币最终状态，求其初始状态。

# 分析

- 考虑直接套用给定的X操作样板，但简单试验后不难发现这是不可能的。
- 根据样板，建立若干方程，求解出A数组，这是很容易做到的。
- 接下来要逆推求出原先的硬币状态，如果用基本的模拟，时间复杂度势必非常高。
- 对于长度为K的硬币序列C的多次操作（逆操作可类似求解）, 有两种方法

# 方法一

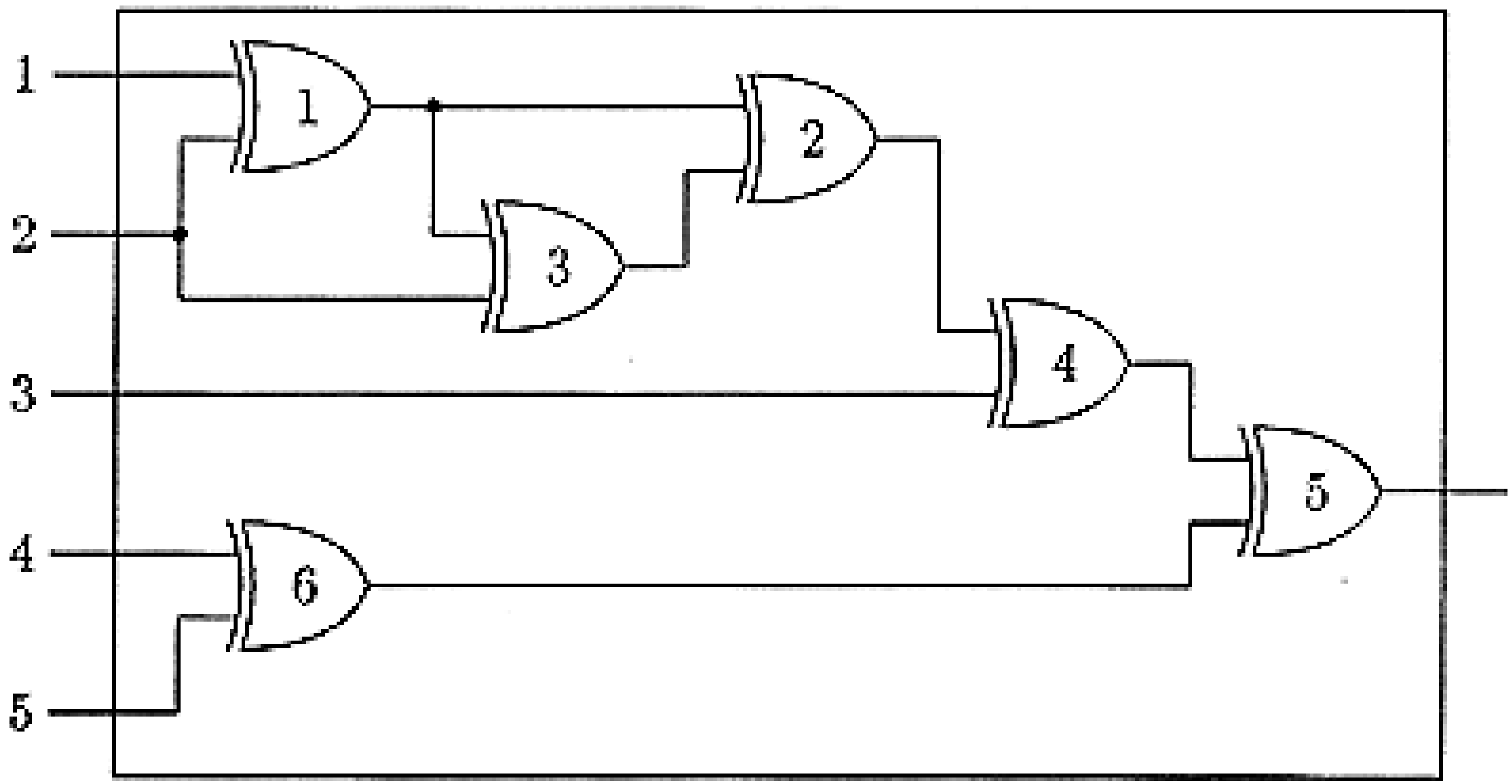
- 设G[i]表示$C_{i+1}..C_k,C_1..C_i$这个硬币序列，经过一次X操作，是否会导致$C_i$变化
- 设q[i][x]代表如果$C_i$变化，那么G[x]是否会发生变化，q数组可以预先处理, 并对其进行位压缩再使用xor进行模拟，降低常数项
- 时间$O(L^3+MD)$, 空间$O(NL+N)$

# 方法二

- 根据A，可以用固定的矩阵B表示X操作。那么$C*B^{Di}$表示对硬币序列的Di次x操作，而矩阵B的$D_i$次方可以用logi次乘法完成
- 时间$O(L^3+MK^3logD)$, 空间$O(NL+N^2logD)$

# 例5. XOR电路

- 输入用长度为n的01串表示。给出A和B，统计A~B中有多少个串让输出为1



# 分析

- 使输出为1的输入满足模线性方程
- 对于给定的上限upper和下限lower，我们可以分别求出在下限是00…0的情况下上限是upper的解数u和上限是lower的解数l。若lower是一组满足条件的解，则总的解数为u-l+1，否则最终的解数是u-l
- 问题的关键：计算上限upper和下限00…0时解的个数

# 分析

- 方程只有1个而变量有多个，设变量个数为m，则可以有m-1个数任意取值，因为最后一个数一定有办法使方程成立
- 设编号最大的输入代表的变量为最后一个变量（设为k），则除k以外的其他输入可以任意取值，而k的取值受到限制

# 例6. 灯泡

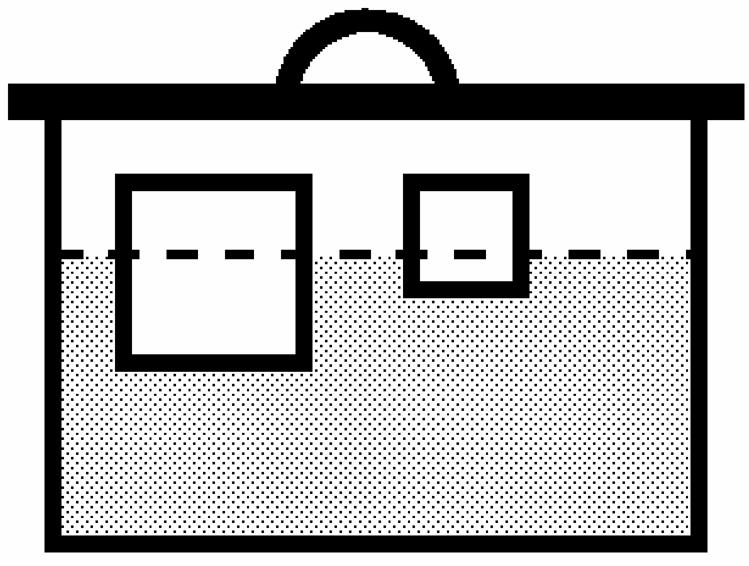
- 有n(<=$10^6$)个灯排列成环形. 每个单位时间, 灯i改变状态当且仅当上一个时刻它下一个灯(i<n时为i+1, i=n时为1)开着.
- 给n个灯的初始状态以及m(m<=$10^9$), 给出m时间单位以后的状态.

# 分析

- 算法一: 直接模拟O(nm)
- 算法二: 事先计算循环节, 则时间复杂度和m无关, 上限是$O(n2^n)$, 但具体运行时间不好估计
- 算法三: 第一次$a_{1,i} = a_{0,i}$ xor $a_{0,i+1}$, 第二次$a_{2,i} = a_{1,i}$ xor $a_{1,i+1} = (a_{0,i}$ xor $a_{0,i+1})$ or $(a_{0,i+1}$ or $a_{0,i+2}) = a_{0,i}$ xor $a_{0,i+2}$, 第四次$a_{4,i}=a_{2,i}$ xor $a_{2,i+2} = (a_{0,i}$ xor $a_{0,i+2})$xor$(a_{0,i+2}$ xor $a_{0,i+4})=a_{0,i}$ xor $a_{0,i+4}$
- 至此,规律已经很明显. 二分法即可. O(nlogm)

# 例7. 水桶

- 桶里注入密度为1的水, 然后放入大小不一的立方体, 最后盖一个盖子并压紧.
- 给出桶底面积S, 高H, 水的体积V(V<=S*H), 以及n个立方体的边长和密度,求最后的水位高度



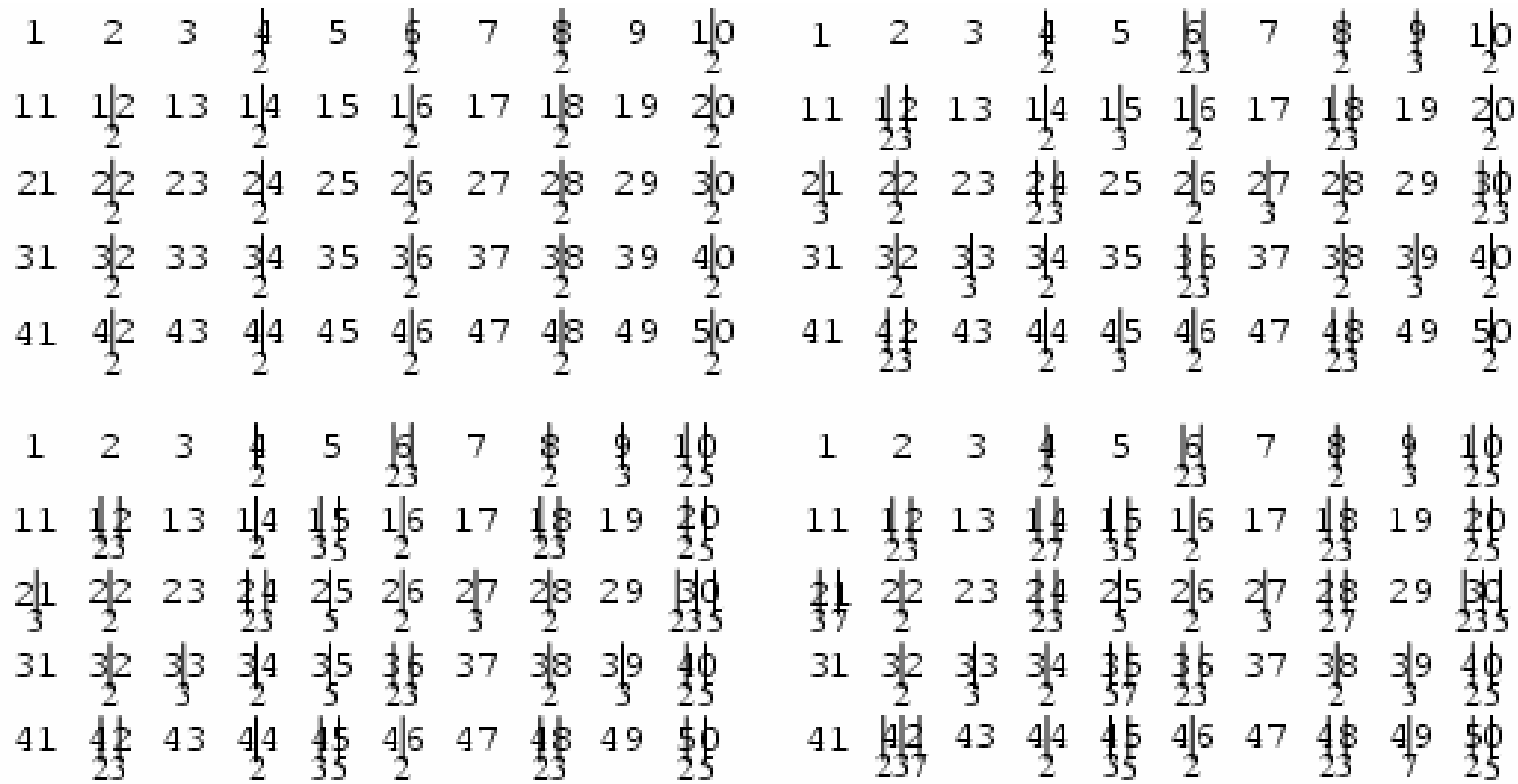
# 和素数有关的问题

- 如何求1~n的所有素数?
- 如何判断一个数n是否为素数?
- 如何求两个数的最大公约数?
- 如何给一个数n分解素因数?

# 问题1: 1~n的素数

- 假设要求1~100的素数
  - 2是素数, 删除2*2, 2*3, 2*4, …, 2*50
  - 第一个没被删除的是3, 删除3*3, 3*4, 3*5,…,3*33
  - 第一个没被删除的是5, 删除5*5, 5*6, … 5*20
- 得到素数p时, 需要删除p*p, p*(p+1), … p*[n/p], 运算量为[n/p]-p, 其中p不超过$n^{1/2}$(想一想, 为什么)

# Eratosthenes的筛子



# 小知识：各种筛法

| *Time* | *Space* | |
|---|---|---|
| $n \log \log n$ | $n$ | Eratosthenes c. 250 BC |
| $n \log \log n$ | $\sqrt{n}$ | [BH77] |
| $n / \log \log n$ | $n / \log \log n$ | [Pri81] |
| $n$ | $\sqrt{n} / \log \log n$ | [Pri83] |
| $\dfrac{n}{\log \log n}$ | $\dfrac{n}{\log n \log \log n}$ | [DJS96] |
| $\dfrac{n \log(\ell + 1)}{\log \log n}$ | $\dfrac{n}{(\log n)^{\ell} \log \log n}$ | For $\ell > 1$, $(\log n)^{\ell} \leq \sqrt{n}$ [Sor98] |
| $n$ | $\dfrac{\sqrt{n}}{(\log \log n)^2}$ | [Sor98] |
| $\dfrac{n^{1+r}}{\log \log n}$ | $n^{1/2-r}$ | For $0 < r < 1/2$ [Sor98] |
| $n / \log \log n$ | $\sqrt{n}$ | [AB04] |
| $n / \log \log n$ | $n^{1/3+\epsilon}$ | [Gal00, Gal04] |
| $n \log n$ | $(\log n)^2$ | Sorenson 2006 |

# 小知识: 素数的个数

- 近似公式(Legendre常数B=-1.08366) $\pi(n) \sim \frac{n}{\ln n + B}$

| $n$ | $\pi(10^n)$ | reference |
|---|---|---|
| 1 | 4 | antiquity |
| 2 | 25 | L. Pisano (1202; Beiler) |
| 3 | 168 | F. van Schooten (1657; Beiler) |
| 4 | 1,229 | F. van Schooten (1657; Beiler) |
| 5 | 9,592 | T. Brancker (1668; Beiler) |
| 6 | 78,498 | A. Felkel (1785; Beiler) |
| 7 | 664,579 | J. P. Kulik (1867; Beiler) |
| 8 | 5,761,455 | Meissel (1871; corrected) |
| 9 | 50,847,534 | Meissel (1886; corrected) |
| 10 | 455,052,511 | Lehmer (1959; corrected) |

# 问题2: 素数判定

- 枚举法: $O(n^{1/2})$, 指数级别
- 改进的枚举法: $O(phi(n^{1/2}))=O(n^{1/2}/logn)$, 仍然是指数级别
- 概率算法: Miller-Rabin测试 + Lucas-Lehmer测试

# Miller-Rabin测试

- 对于奇数n, 记$n=2^r*s+1$, 其中s为奇数
- 随机选a($1<=a<=n-1$), n通过测试的条件是
  - $a^s\equiv 1(\mathrm{mod}\ n)$, 或者
  - 存在$0<=j<=r-1$使得$a^{2^j*s}\equiv -1(\mathrm{mod}\ n)$
- 素数对于所有a通过测试, 合数通过测试的概率不超过1/4
- 只测试**a=2, 3, 5, 7,** 则**$2.5*10^{13}$**以内唯一一个可以通过所有测试的数为**3215031751**

# 思考：区间内的素数

- 给出n, m($n<=10^6$, $m<=10^5$), 求n~n+m之间的素数有多少个
- 哪种方法快? 筛还是依次素数判定?

# 小知识：PRIMES is in P

- 三个印度人M.Agrawal, N.Kayal和N.Saxena与2002年证明了PRIMES可以在多项式时间内完成

```
Input:  integer n > 1
    if ( n is of the form a^b, b > 1 ) output COMPOSITE;
    r = 2;
    while(r < n) {
        if ( gcd(n,r) ≠ 1 ) output COMPOSITE;
        if (r is prime)
            let q be the largest prime factor of r − 1;
            if (q ≥ 4√r log n) and (n^((r−1)/q) ≢ 1 (mod r))
                break;
        r ← r + 1;
    }
    for a = 1 to 2√r log n
        if ( (x − a)^n ≢ (x^n − a) (mod x^r − 1, n)) output COMPOSITE;
    output PRIME;
```

# 关于AKS算法

- Commenting on the impact of this discovery, P. Leyland noted, "One reason for the excitement within the mathematical community is not only does this algorithm settle a long-standing problem, it also does so in a **brilliantly simple manner**. Everyone is now wondering what else has been similarly overlooked" (quoted by Crandall and Papadopoulos 2003).

# 问题3: 最大公约数

- 方法一: 使用惟一分解定理, 先分解素因数, 然后求最大公约数
- 方法二: (Euclid算法)利用公式gcd(a, b)=gcd(b, a mod b), 时间复杂度为O(logb)
- 方法三: (二进制算法) 若a=b, gcd(a,b)=a, 否则
  - A和b均为偶数, gcd(a,b)=2*gcd(a/2,b/2)
  - A为偶数, b为奇数, gcd(a,b)=gcd(a/2,b)
  - 如果a和b均为奇数, gcd(a,b)=gcd(a-b,b)
- 不需要除法, 适合大整数

# 扩展问题

- 一定存在整数x,y，使得ax+by=gcd(a,b)

```
int gcd(int a, int b, int&x, int& y){
  if(!b){ x = 1; y = 0; return a; }
  else{ int r = gcd(b, a%b, x, y);
        t = x; x = y; y = t – a/b*y;
        return r; }
}
```

- 由数学归纳法可证明ax+by=gcd(a,b)
- 满足$ax+by=d$的数对$(x,y)$不是惟一的, 因为当$x$增加$b$且$y$减少$a$时和不变。

# 问题4: 分解因数

- 分解因数可以转换为求最小素因子(找到最小素因子后递归求解)
- 分解素因数后得到惟一分解式sum{$pi^{ki}$}, 可以求出约数个数, 即所有$k_i$+1的乘积(由乘法原理容易证明)
- 方法一: 试除法
- 方法二: pollard-rho算法

# 小知识：因数分解算法

**Overview of Integer Factoring Algorithms:** Input $n$, length $O(\log n)$

| *Algorithm* | *Time* | |
|---|---|---|
| Trial Division | $O(\sqrt{n}/\log n)$ | [Knu98] |
| Fermat's Method | $O(n)$ | [CP01] |
| Lehman's Method | $n^{1/3+o(1)}$ | [CP01] |
| Pollard $\rho$ | $\sqrt{p \log p}$? | [Pol75] |
| Pollard $p-1$ | $\sqrt{n}$ | [Pol74] |
| Pollard-Strassen | $n^{1/4+o(1)}$ | [Mon87] |
| Shanks | $n^{1/5+o(1)}$ | ERH |
| Quadratic Sieve | $\exp[O(\sqrt{\log n \log \log n})]$ | Heur. [Pom82] |
| ECM | $\exp[O(\sqrt{\log p \log \log p})]$ | Heur. [Len87] |
| NFS | $\exp[O((\log n)^{1/3}(\log \log n)^{2/3})]$ | Heur. [LLMP90] |

# RSA加密

**Public-Key:** Encryption and decryption keys are different.

**Setup:**

Choose primes $p$ and $q$, set $n := pq$ and $\phi(n) := (p-1)(q-1)$.

Choose an encryption key $e$ at random, and compute $d := e^{-1} \bmod \phi(n)$.

**Encryption:**

Let $M < n$ be the plaintext message. We assume $\gcd(M, n) = 1$. The ciphertext $C$ is computed as $C := M^e \bmod n$.

**Decryption:**

Compute $M := C^d \bmod n$.

Note: $C^d \equiv (M^e)^d \equiv M^{ed} \equiv M^{k\phi(n)+1} \equiv (M^k)^{\phi(n)} M^1 \equiv M \pmod{n}$.

**Analysis:** The running times for encryption is $O((\log n)^2 \log e)$ and decryption is similarly $O((\log n)^2 \log d)$.

# 小知识：历史上的RSA

| number | digits | prize | factored (references) |
|---|---|---|---|
| RSA-100 | 100 | | Apr. 1991 |
| RSA-110 | 110 | | Apr. 1992 |
| RSA-120 | 120 | | Jun. 1993 |
| RSA-129 | 129 | $100 | Apr. 1994 (Leutwyler 1994, Cipra 1995) |
| RSA-130 | 130 | | Apr. 10, 1996 |
| RSA-140 | 140 | | Feb. 2, 1999 (te Riele 1999a) |
| RSA-150 | 150 | | Apr. 6, 2004 (Aoki 2004) |
| RSA-155 | 155 | | Aug. 22, 1999 (te Riele 1999b, Peterson 1999) |
| RSA-160 | 160 | | Apr. 1, 2003 (Bahr et al. 2003) |
| RSA-200 | 200 | | May 9, 2005 (see Weisstein 2005a) |
| RSA-576 | 174 | $10000 | Dec. 3, 2003 (Franke 2003; see Weisstein 2003) |
| RSA-640 | 193 | $20000 | Nov. 4, 2005 (see Weisstein 2005b) |
| RSA-704 | 212 | $30000 | open |
| RSA-768 | 232 | $50000 | open |
| RSA-896 | 270 | $75000 | open |
| RSA-1024 | 309 | $100000 | open |
| RSA-1536 | 463 | $150000 | open |
| RSA-2048 | 617 | $200000 | open |

# RSA-640 Factored

**By Eric W. Weisstein**

November 8, 2005--A team at the German Federal Agency for Information Technology Security (BSI) recently announced the factorization of the 193-digit number

> 310 7418240490 0437213507 5003588856 7930037346 0228427275 4572016194 8823206440
> 5180815045 5634682967 1723286782 4379162728 3803341547 1073108501 9195485290 0733772482
> 2783525742 3864540146 9173660247 7652346609

known as RSA-640 (Franke 2005). The team responsible for this factorization is the same one that previously factored the 174-digit number known as RSA-576 (*MathWorld* headline news, December 5, 2003) and the 200-digit number known as RSA-200 (*MathWorld* headline news, May 10, 2005).

RSA numbers are composite numbers having exactly two prime factors (i.e., so-called semiprimes) that have been listed in the Factoring Challenge of RSA Security®.

While composite numbers are defined as numbers that can be written as a product of smaller numbers known as factors (for example, 6 = 2 x 3 is composite with factors 2 and 3), prime numbers have no such decomposition (for example, 7 does not have any factors other than 1 and itself). Prime factors therefore represent a fundamental (and unique) decomposition of a given positive integer. RSA numbers are special types of composite numbers particularly chosen to be difficult to factor, and they are identified by the number of digits they contain.

While RSA-640 is a *much* smaller number than the 7,816,230-digit monster Mersenne prime known as $M_{42}$ (which is the largest prime number known), its factorization is significant because of the curious property that proving or disproving a number to be prime ("primality testing") seems to be much easier than actually identifying the factors of a number ("prime factorization"). Thus, while it is trivial to multiply two large numbers $p$ and $q$ together, it can be extremely difficult to determine the factors if only their product $pq$ is given. With some ingenuity, this property can be used to create practical and efficient encryption systems for electronic data.

# 例1. 破解RSA

- 给定M个数，它们的质因子在前T个质数范围内。选出至少一个数, 使得它们的积为完全平方数. 求方案总数

# 分析

- 首先将读入的M个数，分解质因数，并对每个质因数出现次数的奇偶性进行记录。
- 设x[i]=0或1代表是否使用第i个数。可以列出一个关于x[i](1<=i<=m)的位方程组（乘积的所有质因数出现次数均为偶数）。
- 解该方程组，检查最后有多少自变量，设有n个，那么结果应该是$2^n-1$（除去空集）。 时空复杂度均为$O(M^2)$

# 例题2. 奇怪的计数器

- 用如下方式表示数：
- $A_{N-1}*2^{N-1}+A_{N-2}*2^{N-2}+ \ldots +A_1\times 2+A_0$
- 每个A在范围0~2内。
- 初始时所有的A均为0，要处理M次加$2^x$的操作（每个x不一定都相同），每次最多只允许修改4个A的内容。
- 要求模拟这一过程。

# 分析

- 多个2连在一起比较“危险”，容易超过4次的限额
- 让它们之间存在一个0，就缓解了压力
- 考虑这样的限制条件
  - 任意两个相邻的2之间至少有一个0
  - 从最低一位2向更低的位数找，也至少能找到一个0
- 限制条件避免了一次操作需要影响**O(*N*)**个二进制位的退化情况，类似于在排序二叉树中加入了“平衡因子”这个限制。因此不妨称这个限制条件为“平衡性质”。

# 分析

- 一开始的0序列满足平衡性质，因此需要构造从一个平衡状态到另一个平衡状态的过渡法则
- 对于增加$2^i$，考察第i位：
  - 0，那么0->1，考虑这个0所在的两个2之间的区间，如果剩下的都是1（没有0），那么把区间最左边的2进位
  - 1，那么1->0，向前一位进1，如果前一位变成2，注意前一位的前面的区间是否全部都是1，如果那样也要和1)一样修改; 如果前一位原来就是2，那么跳转3)
  - 2，那么2->1，再将其前面一位加1即可

# 例3. 天平

- 有一些砝码, 重量为1, 3, 9, 27, 81…形如3k, 每个重量砝码只有一个. 任意给一个重量为m的物体, 把它放在天平左边, 如何把放置砝码使得天平平衡? 放在左边或者右边都可
- $m<=10^{100}$

# 例4. 987654321问题

- 求有多少个*n*位数平方以后的末9位为987654321。

# 例5. 奇妙的数

- 给定n, m，寻找m位n进制数A，使得2A，3A，…mA的数字均为A数字的排列
- 如m=6, n=10时，142,857是唯一解
- 给定数据最多只有一组解，也可能无解（如m=6, n=100时）

```
2 x 142,857 = 285,714
3 x 142,857 = 428,571
4 x 142,857 = 571,428
5 x 142,857 = 714,285
6 x 142,857 = 857,142
```

# 欧拉定理

- 欧拉函数**: 1~n**中和**n**互素的元素个数$\varphi(n)$
- **Euler**定理 若gcd($a$, $n$)=1则$a^{\varphi(n)} \equiv 1 \pmod{n}$
- 意义：当b很大时$a^b \equiv a^{b \bmod \varphi(n)} \pmod{n}$，让指数一直比较小
- 欧拉函数是积性函数，即当(m,n)=1时f(mn)=f(m)*f(n)

# 思考：欧拉函数的计算

- 给定n，需要多少时间计算$\varphi(n)$？
- 给定n，需要多少时间计算$\varphi(1), \varphi(2), \ldots, \varphi(n)$的所有值？

# 例题：模取幂

- $a$, $p$, $m$, $n$均为正整数，$a$, $p$为素数$1<a, p, m, n<65535$，且$n \le 2a$, $n \le 2p$。求：

$$a^{\overbrace{p^{p^{p^{\cdot^{\cdot^{\cdot^{p}}}}}}}^{m}} \bmod n$$

- 如$a$=32719, $p$=54323, $m$=99, $n$=65399，则结果为46184

# 例题：模取幂

- 记$f(a,p,m,n)$为本题所求的数，
  - $n=1$时，任何数模$n$都是0，所以$f(a,p,m,n)=0$，否则
  - $a=1$时，$a$的任何次幂都是1，结果为$1 \bmod n$；否则
  - $m=0$时，结果为$=a \bmod n$；否则
  - $n=a$时，$a$的次幂永远是$n$的倍数，结果为0；否则
  - $n=2a$时，因为$a, p \geq 2$，如果$a$中有2的因子，则$a$的次幂永远是$n$的倍数，结果为0，否则为$a$；否则
  - $a, n$互素，$f(a,p,m,n)=a^{f(p,p,m-1,\varphi(n))} \bmod n$，问题转变成求$a^k \bmod n$，可以二分求解

# 线性同余方程

- $ax \equiv b (\mathrm{mod}\ n)$
- 方法一：利用Euler函数
  - $a*a^{\varphi(n)-1} \equiv 1 \rightarrow a(b*a^{\varphi(n)-1}) \equiv b$
  - 关键: 求$a^b \mathrm{mod}\ n$
  - $a, a^2, a^4, a^8, a^{16}, \ldots$
  - 同余方程可以做乘法，b做二进制展开选择
- 方法二：扩展的Euclid算法
  - 存在整数$y$，使得$ax-ny=b$
  - 记d=(a,n)，a’=a/d, n’=n/d，必须有d|b
  - a’x-n’y=1*(b/d)
  - 注意：x不唯一, 所有x相差n/d的整数倍

# 中国剩余定理

- 考虑方程组$x\equiv a_i(\text{mod } m_i)$, $m_i$两两互素
- 在$0<=x<M=m_1m_2\ldots m_k$内有唯一解
- 记$M_i=M/m_i$，则$(M_i,m_i)=1$
  - 存在$p_i,q_i$，$M_ip_i+m_iq_i=1$
  - 记录$e_i=M_ip_i$, 则
    - $j=i$时$e_i\equiv 1(\text{mod } m_j)$
    - $j<>i$时$e_i\equiv 0(\text{mod } m_j)$
  - 则$e_1a_1+e_2a_2+\ldots+e_ka_k$就是一个解, 调整得到$[0,m)$内的唯一解（想一想，如何调整）

# 整理一下

- 一般线性方程组$a_i x_i \equiv b_i \pmod{n_i}$
  - $ax \equiv b \pmod{n}$ ➔ $x \equiv b_1 \pmod{n_1}$
  - $x \equiv b_1 \pmod{n_1}$ ➔ $x \equiv b_1 \pmod{p_{1,i}}$
  - 用中国剩余定理
- 其他规则同余方程
  - 二项方程: 借助离散对数(本身??)
  - 高次方程: 分解n, 降幂
  - 单个多变元线性方程: 消法

# 例6. 整数序列

- 已知$\{A_1,\ldots,A_n\}$、B、P求$\{X_1,\ldots,X_n\}$使得
- $A_1X_1+\ldots+A_nX_n = B(\mathrm{mod}\ P)$

# 分析

- 设$g=(A_1,A_2,\ldots A_n,P)$，若g不整除B则无解，否则我们可以用递归构造解：
  - 将$A_1,\ldots A_n$、P和B全部除以g，此时$((A_1,\ldots A_n),P)=1$，
  - 若n=1，则直接求$X_1$使得$A_1X_1 \bmod P=B$；否则
  - 设$(A_1,\ldots,A_{n-1})=D$，则根据欧几里德算法一定存在x和y使得$A_Nx + Dy = B(\bmod p)$，可以令$X_n=x$，则$A_1X_1+\ldots+A_{n-1}X_{n-1}=B-A_nX=Dy(\bmod p)$

# 分析

- $(A_1,\ldots,A_{n-1})=D$, 所以$(A_1,\ldots,A_{n-1},P) = (D,P) \mid (Dy \bmod P)$, 因此完全转化为n-1的情形, 令B=DY mod P即可

# 例1. 电子锁

- 某机要部门安装了电子锁。M个工作人员每人发一张磁卡，卡上有开锁的密码特征。为了确保安全，规定至少要有N(<=M)个人同时使用各自的磁卡才能将锁打开，并且任意N个人在一起都能将锁打开
- 电子锁上至少要有多少种特征？每个人的磁卡至少要有几个特征?

# 分析

- 任意N-1个人在一起，都无法将锁打开，从而必然缺少一种开锁的密码特征A；并且在其余的M-(N-1)个人中，任意一人加入到N-1个人中，他们就能将锁打开
- 结论1: 每M-(N-1)个人拥有一个共同的特征

# 分析

- 设一个N-1人组G所缺少的特征为M(G)
- 结论: 任两个不同的G的M(G)都不同
- 反证法. 设$M(G_1)=M(G_2)=M$. 显然$G_1 \cup G_2$至少包含N个人且他们都缺少特征M，故这些人在一起无法将锁打开，矛盾
- 一共有C(M, N-1)个N-1人组, 因此
- 结论2: 总特征数tot >= C(M, N-1)

# 分析

- 对于每一个工作人员T来说, 在其余M-1个人中任选N-1个人在一起都会因为缺少某种特征而无法开锁，而这缺少的特征必须是T所具备的
- 由于这些缺少的特征各不相同, 所以T的磁卡上至少需要有C(M-1,N-1)个特征

# 分析

- 构造法: 枚举M人选M-(N-1)人的组合, 给它们赋于一个新的共同的特征
- 合法性: 对于每N-1个人, 不具备剩下M-(N-1)拥有的公共特征, 因此打不开门; 由于这N-1个人具备其他所有特征(其他特征在分配时至少分配到了这N-1个人中的一个), 所以随便再找一个就可以打开门
- 最优性: 由结论2, 这个方案是最优的

# 例2. 用AND算XOR

- 对于2n位二进制数A，分成两个n位数后进行AND操作，返回n位数。记该操作为$AND_n(A)$，同理可定义$XOR_m(A)$
- 给定n，求满足m<n的条件下，最大的m值，使得存在
  - 编码映射$g_1$，将2m位数映射为2n位数，和
  - 解码映射$g_2$，将n位数映射为m位数
- 使得$XOR_m(A)=g_2(AND_n(g_1(A)))$

# 分析

- $XOR_m(A)$: 有$2^m$个象，每个都有$2^m$个原象
- $AND_n(A)$: 原象有$3^k$的象有$C(n,k)$个
- 问题转化为：有$2^n$个箱子，其中有$C(n,i)$个容积为$3^i$，求$m(m<n)$的最大值，使得$2^m$个大小为$2^m$的物品可以被放入
- 算法：二分枚举m，算出每一种箱子可以放多少物品，再乘以这种箱子的数量，检测总和是否满足。需要高精度. 时间复杂度：$O(n\log n)$

# 例3. 彩票

- 大街上到处在卖彩票，一元钱一张。购买撕开它上面的锡箔，你会看到一个漂亮的图案。图案有$n$种，如果你收集到所有$n$种彩票，就可以得大奖。请问，在平均情况下，需要买多少张彩票才能得到大奖呢？

# 分析

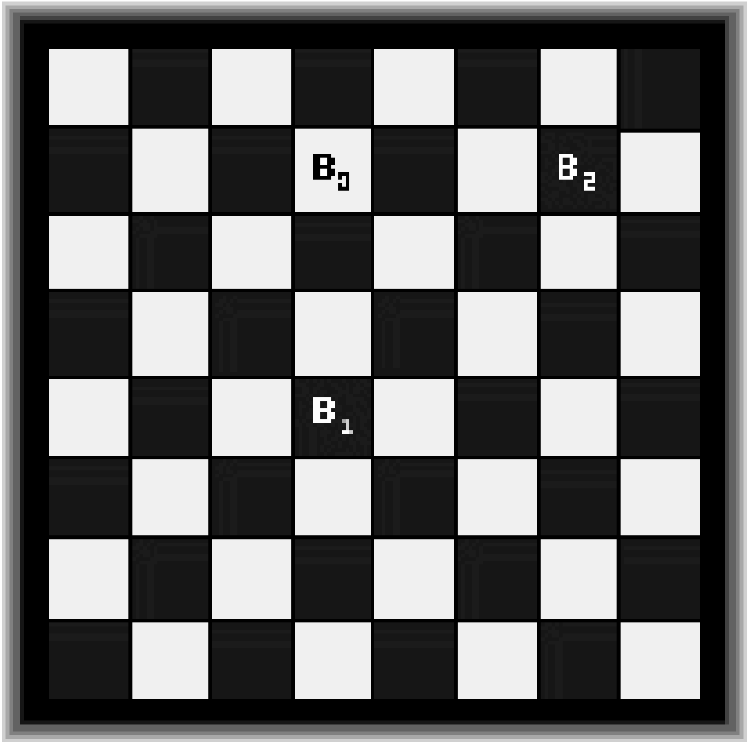
- 已经有k个图案, $s=k/n$, 拿到一个新的需要
  - 1次的概率: $1-s$
  - 2次的概率: $s*(1-s)$
  - t次的概率: $s^{t-1}*(1-s)$
- 期望: 概率加权和
  - $(1-s)*(1 + 2s + 3s^2 + 4s^3 + \ldots) = (1-s)E$
  - 而$sE = s + 2s^2 + 3s^3 + \ldots = E-(1+s+s^2+\ldots)$
  - 移项得$(1-s)E = 1+s+s^2+\ldots=1/(1-s) = n/(n-k)$

# 分析

- 总结
  - 已有0个图案: 拿1次就可以多搜集一个
  - 已有1个图案: 平均拿n/(n-1)次就可多搜集一个
  - 已有k个图案: 平均拿n/(n-k)次就可多搜集一个
- 所以总次数为: n(1+1/2+1/3+…+1/n)

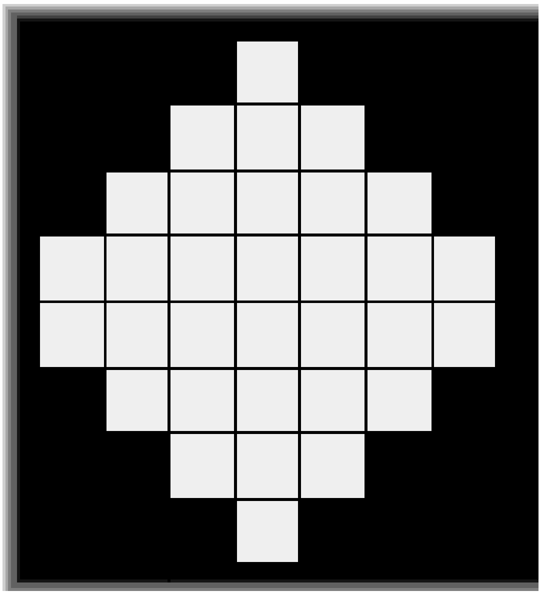
# 互不攻击的象

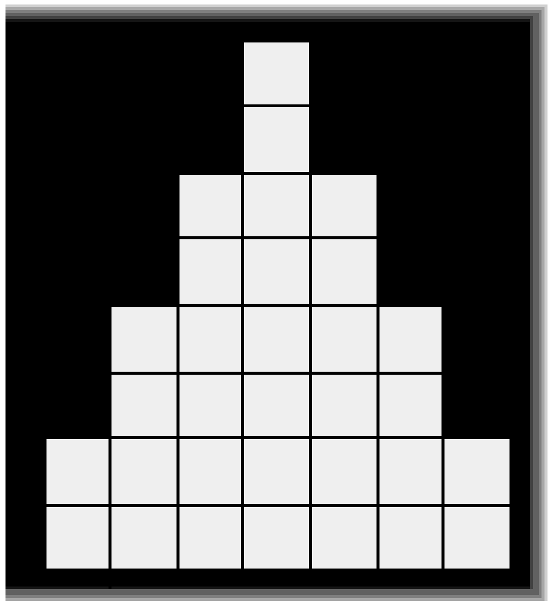
- 两个“象”互不攻击当且仅当它们不处于同一条斜线上。例如B1和B2互相攻击，但是B1和B3、B2和B3都不互相攻击。
- 在一个$n \times n(n \leqslant 30)$的棋盘上放$k$个互不攻击的象有多少种方法？例如，当$n=8$，$k=6$时有5 599 888种方法。



# 分析

- 白格象和黑格象互不相干, 抽白格得到左图
- 行顺序无关, 重排得右图
- 问题: 右图放t个象的方案数?



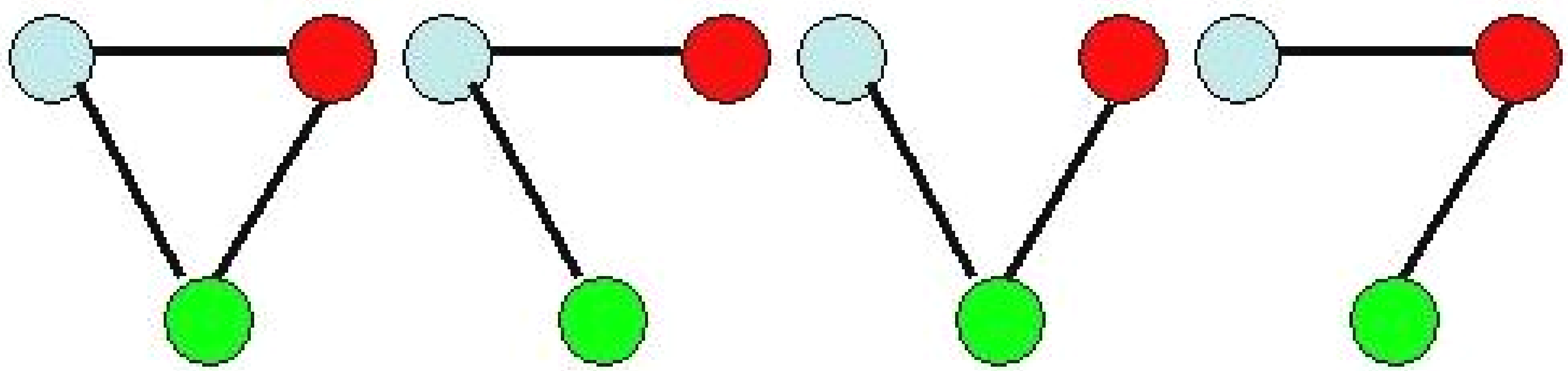


# 分析

- $d_t[i,j]$表示第i行及下面的行放j个象的方案数
- 已经放了t-j个象, 每放一个象都会导致某列被禁止
- 设第i行有c[i]个格子
  - 放: 有c[i]-(t-j)种选择, 转移到$d_t[i+1,j-1]$
  - 不放: 转移到$d_t[i+1,j]$
- 因此$d_t[i,j] = (c[i]-t+j)*d_t[i+1,j-1]+d_t[i+1,j]$
- 枚举白格象个数t, 把所有$d_t[1,t]*d_{k-t}[1,k-t]$加起来即可

# 带标号连通图计数

- 统计有n(<=50)个顶点的连通图有多少个



- n=3时有四个不同的图
- n=4时有38个图

# 分析

- 设f(n)为所求答案, g(n)为有n个顶点的非连通图, 则$f(n)+g(n)=h(n)=2^{n(n-1)/2}$.
- g(n)可以这样计算: 先枚举1所在连通分量的情况. 假设共有k个点就有C(n-1,k-1)种集合. 确定集合后, 第一连通分量有f(k)种情况, 其他连通分量有h(n-k)种情况，因此答案是
- g(n)=sum{C(n-1,k-1)*f(k)*h(n-k)}, k=1..n-1
- 状态有n个, 决策n个, 时间复杂度为$O(n^2)$
- 每次计算出g(n)时立刻需要算出f(n)和h(n)

# 分析

- f(n)=1,1,4,38,728,26704,1866256,251548592
- h(n)=1,2,8,64,1024,
- 以f(5)为例. 显然h(n)=1,2,8,64,1024…先算g(5)
  - C(4,0)*f(1)*h(4)=1*1*64=64
  - C(4,1)*f(2)*h(3)=4*1*8=32
  - C(4,2)*f(3)*h(2)=6*4*2=48
  - C(4,3)*f(4)*h(1)=4*38*1=152
- 则f(5)=h(5)-g(5)=1024-64-32-48-152=728